

[SREES] SeminarSKI - SystemModel Dokumentacija

Generated by Doxygen 1.9.4

1 Namespace Index	1
1.1 Namespace List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 SystemModel Namespace Reference	7
4.1.1 Typedef Documentation	7
4.1.1.1 AdmittanceMatrix	8
4.1.1.2 Branch	8
4.1.1.3 dfidx	8
4.1.1.4 fi	8
4.1.2 Enumeration Type Documentation	8
4.1.2.1 ThreePhaseLoadConfigurationsType	8
4.1.2.2 TypeOfBranch	8
4.1.2.3 TypeOfBus	9
4.1.3 Function Documentation	9
4.1.3.1 operator<<()	9
5 Class Documentation	11
5.1 SystemModel::Bus Class Reference	11
5.1.1 Constructor & Destructor Documentation	12
5.1.1.1 Bus()	12
5.1.2 Member Function Documentation	12
5.1.2.1 getActivePower()	12
5.1.2.2 getReactivePower()	13
5.1.2.3 getTypeOfBus()	13
5.1.2.4 getVoltageMagnitude()	14
5.1.2.5 getVoltagePhase()	14
5.1.2.6 setActivePower()	15
5.1.2.7 setReactivePower()	15
5.1.2.8 setVoltageMagnitude()	16
5.1.2.9 setVoltagePhase()	16
5.2 SystemModel::SystemModel Class Reference	17
5.2.1 Constructor & Destructor Documentation	18
5.2.1.1 SystemModel()	18
5.2.2 Member Function Documentation	18
5.2.2.1 addBus()	18
5.2.2.2 addCapacitorBank()	19
5.2.2.3 addGenerator()	19

5.2.2.4 addLine()	20
5.2.2.5 addLoad()	20
5.2.2.6 addSlackGenerator()	21
5.2.2.7 addTransformer()	21
5.2.2.8 changeCapacitorBank()	22
5.2.2.9 changeLine()	22
5.2.2.10 changeTransformer()	23
5.2.2.11 getAdmittanceMatrix()	23
5.2.2.12 getBranches()	24
5.2.2.13 getBus()	24
5.2.2.14 getBusFunctions()	25
5.2.2.15 getCapacitorBanks()	26
5.2.2.16 getDerivativesOfBusFunctions()	26
5.2.2.17 getNumberOfBuses()	27
5.2.2.18 hasSlackBeenAssigned()	27
5.2.2.19 removeBranch()	28
5.2.2.20 removeBus()	28
5.2.2.21 removeCapacitorBank()	28
5.2.3 Friends And Related Function Documentation	28
5.2.3.1 operator<<	28
6 File Documentation	31
6.1 export.cpp File Reference	31
6.1.1 Function Documentation	32
6.1.1.1 exportToHTML()	32
6.1.1.2 exportToLatex()	33
6.1.1.3 exportToTxt()	35
6.1.2 Variable Documentation	36
6.1.2.1 eps	36
6.2 export.h File Reference	37
6.2.1 Function Documentation	37
6.2.1.1 exportToHTML()	38
6.2.1.2 exportToLatex()	39
6.2.1.3 exportToTxt()	41
6.3 export.h	42
6.4 import.cpp File Reference	43
6.4.1 Function Documentation	43
6.4.1.1 importFromTxt()	43
6.5 import.h File Reference	45
6.5.1 Function Documentation	46
6.5.1.1 importFromTxt()	46
6.6 import.h	48

6.7 main.cpp File Reference	48
6.7.1 Function Documentation	48
6.7.1.1 main()	49
6.7.2 Variable Documentation	49
6.7.2.1 eps	49
6.8 newtonRaphson.cpp File Reference	50
6.8.1 Function Documentation	51
6.8.1.1 absVector()	51
6.8.1.2 adjoint()	51
6.8.1.3 cofactor()	52
6.8.1.4 determinant()	53
6.8.1.5 inverseMatrix()	54
6.8.1.6 newtonRaphson()	55
6.8.1.7 operator*()	56
6.8.1.8 operator-()	56
6.9 newtonRaphson.h File Reference	57
6.9.1 Function Documentation	58
6.9.1.1 absVector()	58
6.9.1.2 adjoint()	58
6.9.1.3 cofactor()	59
6.9.1.4 determinant()	60
6.9.1.5 inverseMatrix()	61
6.9.1.6 newtonRaphson()	62
6.9.1.7 operator*()	63
6.9.1.8 operator-()	63
6.10 newtonRaphson.h	64
6.11 systemModel.cpp File Reference	64
6.11.1 Macro Definition Documentation	64
6.11.1.1 PI	65
6.12 systemModel.h File Reference	65
6.13 systemModel.h	66
Index	69

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

SystemModel	7
---------------------------------------	---

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

SystemModel::Bus	11
SystemModel::SystemModel	17

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

export.cpp	31
export.h	37
import.cpp	43
import.h	45
main.cpp	48
newtonRaphson.cpp	50
newtonRaphson.h	57
systemModel.cpp	64
systemModel.h	65

Chapter 4

Namespace Documentation

4.1 SystemModel Namespace Reference

Classes

- class [Bus](#)
- class [SystemModel](#)

Typedefs

- using [fi](#) = std::pair< std::function< double(std::vector< double >)>, std::function< double(std::vector< double >)>> >
- using [dfidx](#) = std::pair< std::vector< std::function< double(std::vector< double >)> >, std::vector< std::function< double(std::vector< double >)> > >
- using [AdmittanceMatrix](#) = std::vector< std::tuple< uint8_t, uint8_t, std::complex< double > > >
- using [Branch](#) = std::tuple< [TypeOfBranch](#), uint8_t, uint8_t, double, double, double, double >

Enumerations

- enum class [TypeOfBus](#) { [Slack](#) , [PV](#) , [PQ](#) }
- enum class [ThreePhaseLoadConfigurationsType](#) { [Star](#) , [GroundedStar](#) , [Delta](#) }
- enum class [TypeOfBranch](#) { [Line](#) , [Transformer](#) }

Functions

- std::ostream & [operator<<](#) (std::ostream &stream, const [SystemModel](#) &systemModel)
Output stream operator overload

4.1.1 Typedef Documentation

4.1.1.1 AdmittanceMatrix

```
using SystemModel::AdmittanceMatrix = typedef std::vector<std::tuple<uint8_t, uint8_t, std::complex<double>>>
```

4.1.1.2 Branch

```
using SystemModel::Branch = typedef std::tuple<TypeOfBranch, uint8_t, uint8_t, double, double, double, double>
```

4.1.1.3 dfidx

```
using SystemModel::dfidx = typedef std::pair<std::vector<std::function<double(std::vector<double>)>>, std::vector<std::function<double(std::vector<double>)>>>
```

4.1.1.4 fi

```
using SystemModel::fi = typedef std::pair<std::function<double(std::vector<double>)>, std::function<double(std::vector<double>)>>
```

4.1.2 Enumeration Type Documentation

4.1.2.1 ThreePhaseLoadConfigurationsType

```
enum class SystemModel::ThreePhaseLoadConfigurationsType [strong]
```

Enumerator

Star	
GroundedStar	
Delta	

4.1.2.2 TypeOfBranch

```
enum class SystemModel::TypeOfBranch [strong]
```

Enumerator

Line	
Transformer	

4.1.2.3 TypeOfBus

```
enum class SystemModel::TypeOfBus [strong]
```

Enumerator

Slack	
PV	
PQ	

4.1.3 Function Documentation

4.1.3.1 operator<<()

```
std::ostream & SystemModel::operator<< (  
    std::ostream & stream,  
    const SystemModel & systemModel )
```

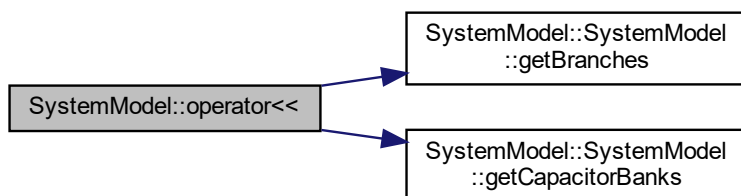
Output stream operator overload

Parameters

<i>stream</i>	Output stream object
<i>systemModel</i>	SystemModel object to be printed to the stream

Returns

Here is the call graph for this function:



Chapter 5

Class Documentation

5.1 SystemModel::Bus Class Reference

```
#include <systemModel.h>
```

Collaboration diagram for SystemModel::Bus:

SystemModel::Bus
<div><div>+ Bus()</div><div>+ getTypeOfBus()</div><div>+ setVoltageMagnitude()</div><div>+ setVoltagePhase()</div><div>+ setActivePower()</div><div>+ setReactivePower()</div><div>+ getVoltageMagnitude()</div><div>+ getVoltagePhase()</div><div>+ getActivePower()</div><div>+ getReactivePower()</div></div>

Public Member Functions

- [Bus](#) ([TypeOfBus](#) typeOfBus)
- [TypeOfBus](#) [getTypeOfBus](#) () const
- void [setVoltageMagnitude](#) (double voltageMagnitude)
Sets the value at which the voltage amplitude for the given bus should be maintained.
- void [setVoltagePhase](#) (double voltagePhase)
Sets the value at which the voltage phase for the given bus should be maintained.
- void [setActivePower](#) (double activePower)

- Sets the value at which the active power for the given bus should be maintained.*
- void [setReactivePower](#) (double reactivePower)
- Sets the value at which the rective power for the given bus should be maintained.*
- std::optional< double > [getVoltageMagnitude](#) () const
- Gets the value at which the voltage magnitude for the given bus should be maintained.*
- std::optional< double > [getVoltagePhase](#) () const
- Gets the value at which the voltage phase for the given bus should be maintained.*
- std::optional< double > [getActivePower](#) () const
- Gets the value at which the active power for the given bus should be maintained.*
- std::optional< double > [getReactivePower](#) () const
- Gets the value at which the rective power for the given bus should be maintained.*

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Bus()

```
SystemModel::Bus::Bus (
    TypeOfBus typeOfBus ) [inline]
```

5.1.2 Member Function Documentation

5.1.2.1 getActivePower()

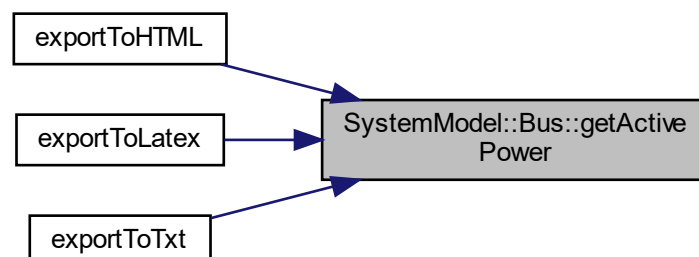
```
std::optional< double > SystemModel::Bus::getActivePower ( ) const
```

Gets the value at which the active power for the given bus should be maintained.

Returns

Value of active power for the bus

Here is the caller graph for this function:



5.1.2.2 getReactivePower()

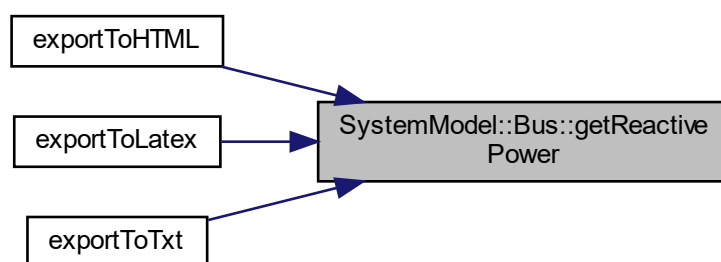
```
std::optional< double > SystemModel::Bus::getReactivePower ( ) const
```

Gets the value at which the rective power for the given bus should be maintained.

Returns

Value of reactive power for the bus

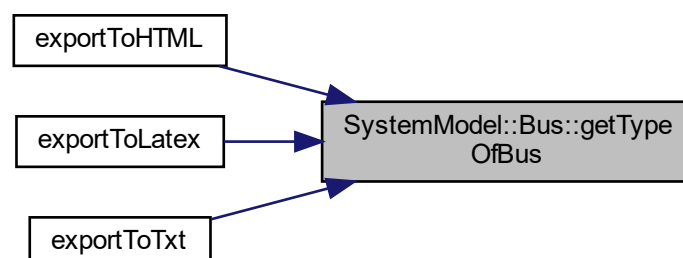
Here is the caller graph for this function:



5.1.2.3 getTypeOfBus()

```
TypeOfBus SystemModel::Bus::getTypeOfBus ( ) const [inline]
```

Here is the caller graph for this function:



5.1.2.4 getVoltageMagnitude()

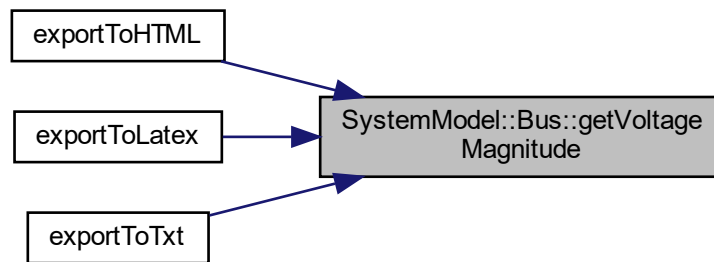
```
std::optional< double > SystemModel::Bus::getVoltageMagnitude ( ) const
```

Gets the value at which the voltage magnitude for the given bus should be maintained.

Returns

Value of voltage magnitude of the bus

Here is the caller graph for this function:



5.1.2.5 getVoltagePhase()

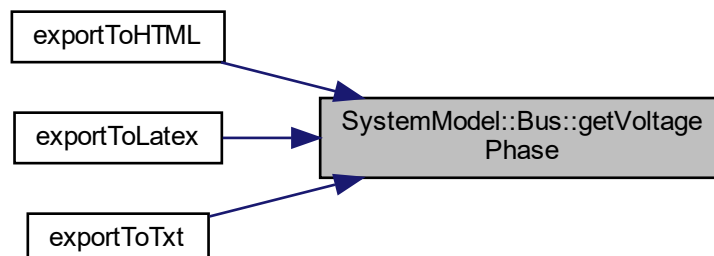
```
std::optional< double > SystemModel::Bus::getVoltagePhase ( ) const
```

Gets the value at which the voltage phase for the given bus should be maintained.

Returns

Value of voltage phase of the bus

Here is the caller graph for this function:



5.1.2.6 setActivePower()

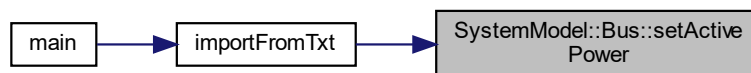
```
void SystemModel::Bus::setActivePower (
    double activePower )
```

Sets the value at which the active power for the given bus should be maintained.

Parameters

<i>activePower</i>	Value of active power for the bus
--------------------	-----------------------------------

Here is the caller graph for this function:



5.1.2.7 setReactivePower()

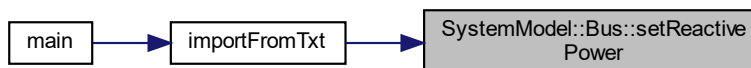
```
void SystemModel::Bus::setReactivePower (
    double reactivePower )
```

Sets the value at which the rective power for the given bus should be maintained.

Parameters

<i>reactivePower</i>	Value of reactive power for the bus
----------------------	-------------------------------------

Here is the caller graph for this function:



5.1.2.8 setVoltageMagnitude()

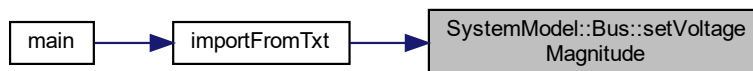
```
void SystemModel::Bus::setVoltageMagnitude (
    double voltageMagnitude )
```

Sets the value at which the voltage amplitude for the given bus should be maintained.

Parameters

<i>voltageMagnitude</i>	Value of voltage magnitude of the bus
-------------------------	---------------------------------------

Here is the caller graph for this function:



5.1.2.9 setVoltagePhase()

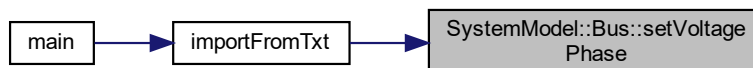
```
void SystemModel::Bus::setVoltagePhase (
    double voltagePhase )
```

Sets the value at which the voltage phase for the given bus should be maintained.

Parameters

<i>voltagePhase</i>	Value of voltage phase of the bus
---------------------	-----------------------------------

Here is the caller graph for this function:



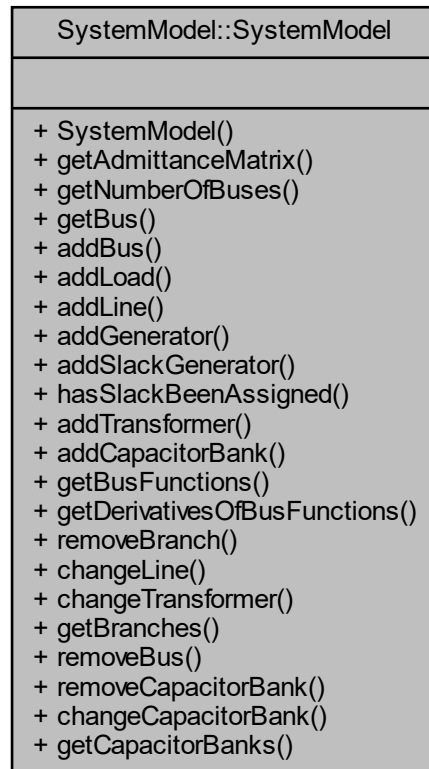
The documentation for this class was generated from the following files:

- [systemModel.h](#)
- [systemModel.cpp](#)

5.2 SystemModel::SystemModel Class Reference

```
#include <systemModel.h>
```

Collaboration diagram for SystemModel::SystemModel:



Public Member Functions

- [SystemModel](#) (uint8_t maxNumberOfBuses)
- [AdmittanceMatrix](#) [getAdmittanceMatrix](#) () const
- uint8_t [getNumberOfBuses](#) () const
- [Bus](#) & [getBus](#) (uint8_t busNumber)
Gets the bus with the given bus number
- void [addBus](#) ([TypeOfBus](#) typeOfBus)
Adds a bus to the system
- void [addLoad](#) (uint8_t busNumber, double activePower, double reactivePower)
Adds a load to a bus
- void [addLine](#) (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b)
Adds a line between buses
- void [addGenerator](#) (uint8_t busNumber, double voltageMagnitude, double activePower)
Adds a generator to a bus
- void [addSlackGenerator](#) (uint8_t busNumber, double voltageMagnitude, double voltagePhase)

- Adds a generator to the slack bus*
 - bool `hasSlackBeenAssigned` () const
Check whether the slack bus has been assigned
- void `addTransformer` (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g, double b)
Adds a transformer between buses
- void `addCapacitorBank` (uint8_t busNumber, double b, `ThreePhaseLoadConfigurationsType` configuration↵
Type)
- *Adds a capacitor bank to a bus*
- fi `getBusFunctions` (uint8_t busNumber) const
Gets the bus functions (fi_P and fi_Q) for the desired bus
- dfidx `getDerivativesOfBusFunctions` (uint8_t busNumber) const
Gets the derivates of the bus functions (dfi_P/dx and dfi_Q/dx) for the desired bus (two rows of the Jacobian associated with the given bus functions)
- void `removeBranch` (uint8_t busNumber1, uint8_t busNumber2)
Removes a line or transformer between buses
- void `changeLine` (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b)
Changes the parameters of the line between buses
- void `changeTransformer` (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g, double b)
Changes the parameters of the transformer between buses
- std::vector< `Branch` > `getBranches` () const
- void `removeBus` (uint8_t busNumber)
Removes the given bus from the system
- void `removeCapacitorBank` (uint8_t busNumber)
Removes the capacitor bank that is connected to the given bus
- void `changeCapacitorBank` (uint8_t busNumber, double b, `ThreePhaseLoadConfigurationsType` configuration↵
Type)
Changes the parameters of the capacitor bank connected to the given bus
- std::vector< std::tuple< uint8_t, double, `ThreePhaseLoadConfigurationsType` > > `getCapacitorBanks` () const

Friends

- std::ostream & `operator<<` (std::ostream &stream, const `SystemModel` &systemModel)

5.2.1 Constructor & Destructor Documentation

5.2.1.1 SystemModel()

```
SystemModel::SystemModel::SystemModel (
    uint8_t maxNumberOfBuses ) [inline]
```

5.2.2 Member Function Documentation

5.2.2.1 addBus()

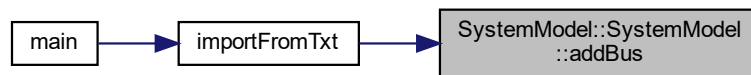
```
void SystemModel::SystemModel::addBus (
    TypeOfBus typeOfBus )
```

Adds a bus to the system

Parameters

<i>typeOfBus</i>	Type of the bus (Slack, PV, PQ) to be added to the system
------------------	---

Here is the caller graph for this function:



5.2.2.2 addCapacitorBank()

```

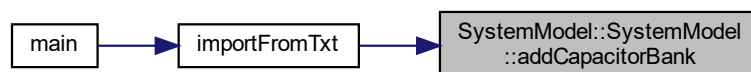
void SystemModel::SystemModel::addCapacitorBank (
    uint8_t busNumber,
    double b,
    ThreePhaseLoadConfigurationsType configurationType )
  
```

Adds a capacitor bank to a bus

Parameters

<i>busNumber</i>	Ordinal number of the desired bus
<i>b</i>	One phase susceptance of the bank
<i>configurationType</i>	Three phase load configuration type (delta, star, grounded star) of the bank

Here is the caller graph for this function:



5.2.2.3 addGenerator()

```

void SystemModel::SystemModel::addGenerator (
    uint8_t busNumber,
  
```

```
double voltageMagnitude,
double activePower )
```

Adds a generator to a bus

Parameters

<i>busNumber</i>	Ordinal number of the desired bus
<i>voltageMagnitude</i>	Voltage magnitude on which the given bus should be maintained

<param name="activePower"Active power on which the given bus should be maintained>

5.2.2.4 addLine()

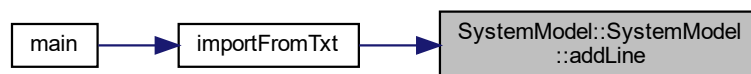
```
void SystemModel::SystemModel::addLine (
    uint8_t busNumber1,
    uint8_t busNumber2,
    double r,
    double x,
    double b )
```

Adds a line between buses

Parameters

<i>busNumber1</i>	Ordinal number of the first bus
<i>busNumber2</i>	Ordinal number of the second bus
<i>r</i>	Series resistance of the transmission line PI equivalent
<i>x</i>	Series reactance of the transmission line PI equivalent
<i>b</i>	Shunt susceptance of the transmission line PI equivalent

Here is the caller graph for this function:



5.2.2.5 addLoad()

```
void SystemModel::SystemModel::addLoad (
    uint8_t busNumber,
    double activePower,
    double reactivePower )
```

Adds a load to a bus

Parameters

<i>busNumber</i>	Ordinal number of the desired bus
<i>activePower</i>	Active power drawn by the load
<i>reactivePower</i>	Reactive power drawn by the load

5.2.2.6 addSlackGenerator()

```
void SystemModel::SystemModel::addSlackGenerator (
    uint8_t busNumber,
    double voltageMagnitude,
    double voltagePhase )
```

Adds a generator to the slack bus

Parameters

<i>busNumber</i>	Ordinal number of the desired bus
<i>voltageMagnitude</i>	Voltage magnitude on which the given bus should be maintained
<i>voltagePhase</i>	Voltage phase on which the given bus should be maintained

5.2.2.7 addTransformer()

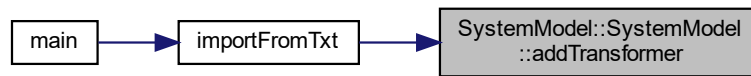
```
void SystemModel::SystemModel::addTransformer (
    uint8_t busNumber1,
    uint8_t busNumber2,
    double r,
    double x,
    double g,
    double b )
```

Adds a transformer between buses

Parameters

<i>busNumber1</i>	Ordinal number of the first bus
<i>busNumber2</i>	Ordinal number of the second bus
<i>r</i>	Series resistance of the transformer PI equivalent
<i>x</i>	Series reactance of the transformer PI equivalent
<i>g</i>	Shunt conductance of the transformer PI equivalent
<i>b</i>	Shunt susceptance of the transformer PI equivalent

Here is the caller graph for this function:



5.2.2.8 changeCapacitorBank()

```

void SystemModel::SystemModel::changeCapacitorBank (
    uint8_t busNumber,
    double b,
    ThreePhaseLoadConfigurationsType configurationType )
  
```

Changes the parameters of the capacitor bank connected to the given bus

Parameters

<i>busNumber</i>	Ordinal number of the desired bus
<i>b</i>	One phase susceptance of the bank
<i>configurationType</i>	Three phase load configuration type (delta, star, grounded star) of the bank

5.2.2.9 changeLine()

```

void SystemModel::SystemModel::changeLine (
    uint8_t busNumber1,
    uint8_t busNumber2,
    double r,
    double x,
    double b )
  
```

Changes the parameters of the line between buses

Parameters

<i>busNumber1</i>	Ordinal number of the first bus
<i>busNumber2</i>	Ordinal number of the second bus
<i>r</i>	Series resistance of the transmission line PI equivalent
<i>x</i>	Series reactance of the transmission line PI equivalent
<i>b</i>	Shunt susceptance of the transmission line PI equivalent

5.2.2.10 changeTransformer()

```
void SystemModel::SystemModel::changeTransformer (
    uint8_t busNumber1,
    uint8_t busNumber2,
    double r,
    double x,
    double g,
    double b )
```

Changes the parameters of the transformer between buses

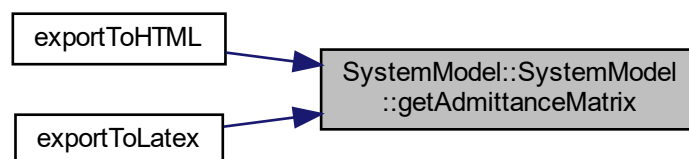
Parameters

<i>busNumber1</i>	Ordinal number of the first bus
<i>busNumber2</i>	Ordinal number of the second bus
<i>r</i>	Series resistance of the transformer PI equivalent
<i>x</i>	Series reactance of the transformer PI equivalent
<i>g</i>	Shunt conductance of the transformer PI equivalent
<i>b</i>	Shunt susceptance of the transformer PI equivalent

5.2.2.11 getAdmittanceMatrix()

```
AdmittanceMatrix SystemModel::SystemModel::getAdmittanceMatrix ( ) const [inline]
```

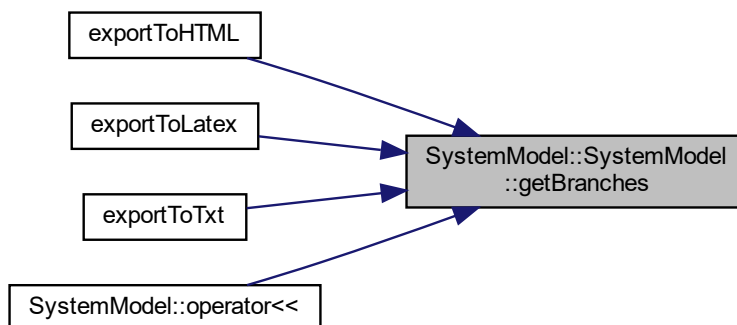
Here is the caller graph for this function:



5.2.2.12 getBranches()

```
std::vector< Branch > SystemModel::SystemModel::getBranches ( ) const [inline]
```

Here is the caller graph for this function:



5.2.2.13 getBus()

```
SystemModel::Bus & SystemModel::SystemModel::getBus (
    uint8_t busNumber )
```

Gets the bus with the given bus number

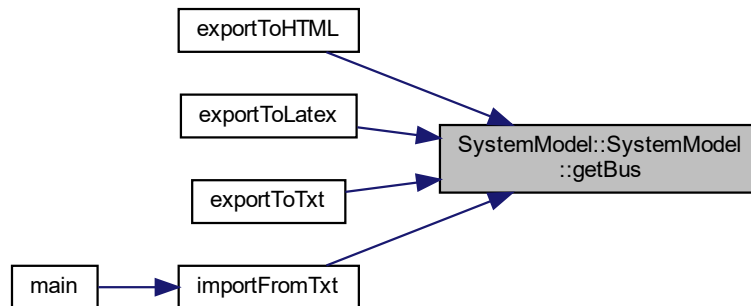
Parameters

<i>busNumber</i>	Ordinal number of the desired bus
------------------	-----------------------------------

Returns

Bus with the given bus number

Here is the caller graph for this function:



5.2.2.14 getBusFunctions()

```
SystemModel::fi SystemModel::SystemModel::getBusFunctions (
    uint8_t busNumber ) const
```

Gets the bus functions (fi_P and fi_Q) for the desired bus

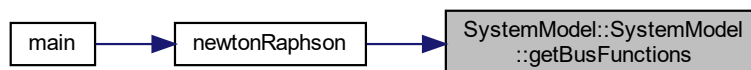
Parameters

<i>busNumber</i>	Ordinal number of the desired bus
------------------	-----------------------------------

Returns

Bus functions for the given bus in the form of `std::pair` of functions, where both functions have a `std::vector` of doubles as parameters and return a double

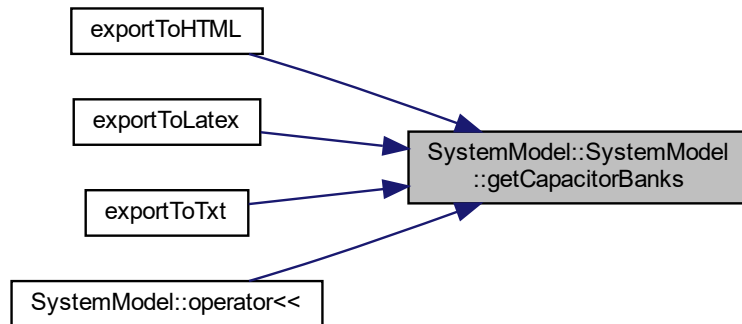
Here is the caller graph for this function:



5.2.2.15 getCapacitorBanks()

```
std::vector< std::tuple< uint8_t, double, ThreePhaseLoadConfigurationsType > > SystemModel↔
::SystemModel::getCapacitorBanks ( ) const [inline]
```

Here is the caller graph for this function:



5.2.2.16 getDerivativesOfBusFunctions()

```
SystemModel::dfidx SystemModel::SystemModel::getDerivativesOfBusFunctions (
    uint8_t busNumber ) const
```

Gets the derivates of the bus functions ($\text{d}f_i_P/\text{d}x$ and $\text{d}f_i_Q/\text{d}x$) for the desired bus (two rows of the Jacobian associated with the given bus functions)

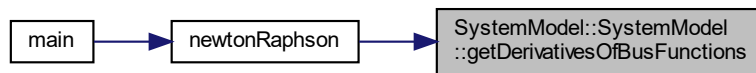
Parameters

<i>busNumber</i>	Ordinal number of the desired bus
------------------	-----------------------------------

Returns

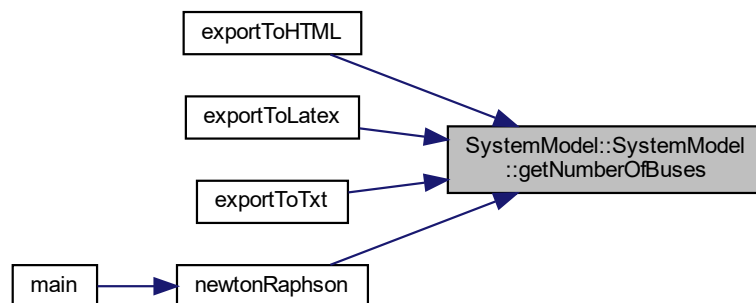
Derivatives of the bus functions for the given bus in the form of std::pair of std::vector-s of functions, where both functions have a std::vector of doubles as parameters and return a double

Here is the caller graph for this function:

**5.2.2.17 getNumberOfBuses()**

```
uint8_t SystemModel::SystemModel::getNumberOfBuses ( ) const [inline]
```

Here is the caller graph for this function:

**5.2.2.18 hasSlackBeenAssigned()**

```
bool SystemModel::SystemModel::hasSlackBeenAssigned ( ) const
```

Check whether the slack bus has been assigned

Returns

True if the slack bus has been assigned and false otherwise

5.2.2.19 removeBranch()

```
void SystemModel::SystemModel::removeBranch (
    uint8_t busNumber1,
    uint8_t busNumber2 )
```

Removes a line or transformer between buses

Parameters

<i>busNumber1</i>	Ordinal number of the first bus
<i>busNumber2</i>	Ordinal number of the second bus

5.2.2.20 removeBus()

```
void SystemModel::SystemModel::removeBus (
    uint8_t busNumber )
```

Removes the given bus from the system

Parameters

<i>busNumber</i>	Ordinal number of the desired bus
------------------	-----------------------------------

5.2.2.21 removeCapacitorBank()

```
void SystemModel::SystemModel::removeCapacitorBank (
    uint8_t busNumber )
```

Removes the capacitor bank that is connected to the given bus

Parameters

<i>busNumber</i>	Ordinal number of the desired bus
------------------	-----------------------------------

5.2.3 Friends And Related Function Documentation

5.2.3.1 operator<<

```
std::ostream & operator<< (
    std::ostream & stream,
    const SystemModel & systemModel ) [friend]
```

The documentation for this class was generated from the following files:

- [systemModel.h](#)
- [systemModel.cpp](#)

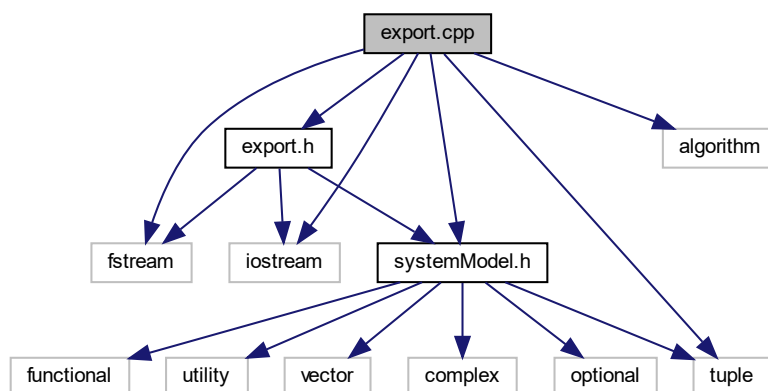
Chapter 6

File Documentation

6.1 export.cpp File Reference

```
#include "export.h"  
#include "systemModel.h"  
#include <algorithm>  
#include <iostream>  
#include <fstream>  
#include <tuple>
```

Include dependency graph for export.cpp:



Functions

- void `exportToLatex` (`SystemModel::SystemModel s`)
Exports `SystemModel` to the `main.tex` file
- void `exportToHTML` (`SystemModel::SystemModel s`)
Exports `SystemModel` to the `main.html` file
- void `exportToTxt` (`const char *filename`, `SystemModel::SystemModel s`)
Exports `SystemModel` to the `.tex` file

Variables

- const double `eps` { 1e-10 }

6.1.1 Function Documentation

6.1.1.1 `exportToHTML()`

```
void exportToHTML (
    SystemModel::SystemModel s )
```

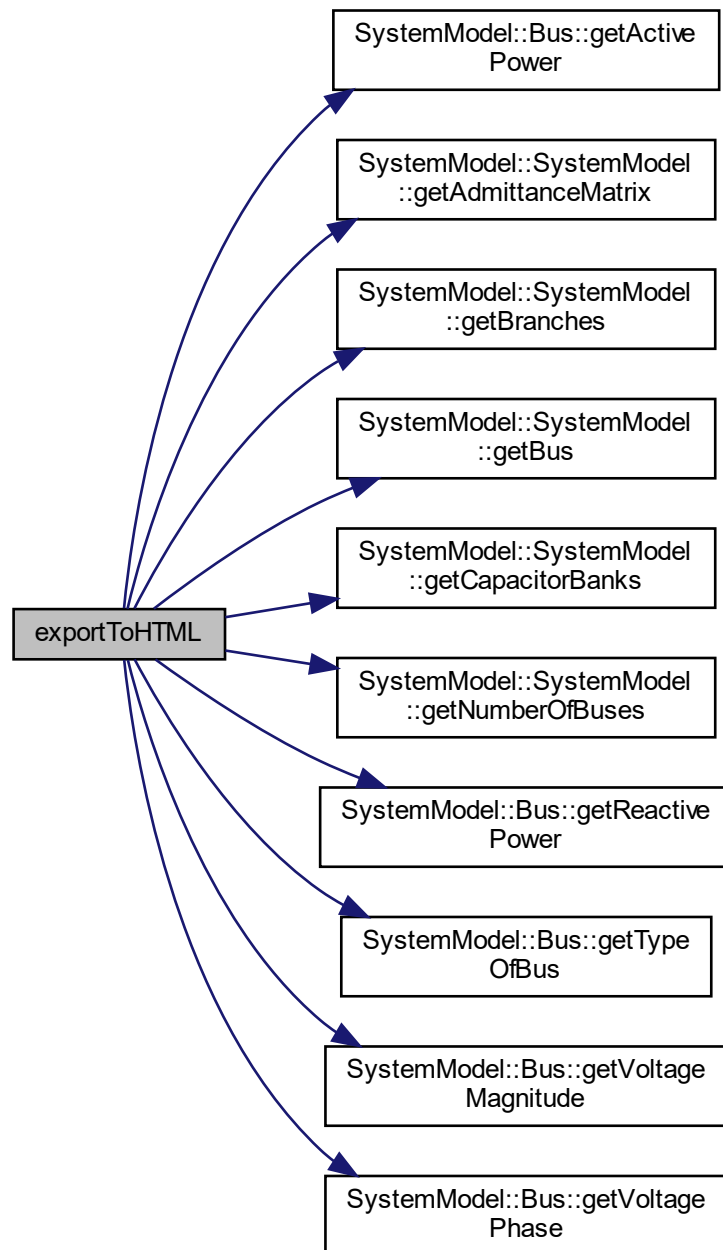
Exports `SystemModel` to the main.html file

Parameters

<code><i>SystemModel::SystemModel</i></code>	System model
--	--------------

Returns

Here is the call graph for this function:



6.1.1.2 exportToLatex()

```
void exportToLatex (  
    SystemModel::SystemModel s )
```

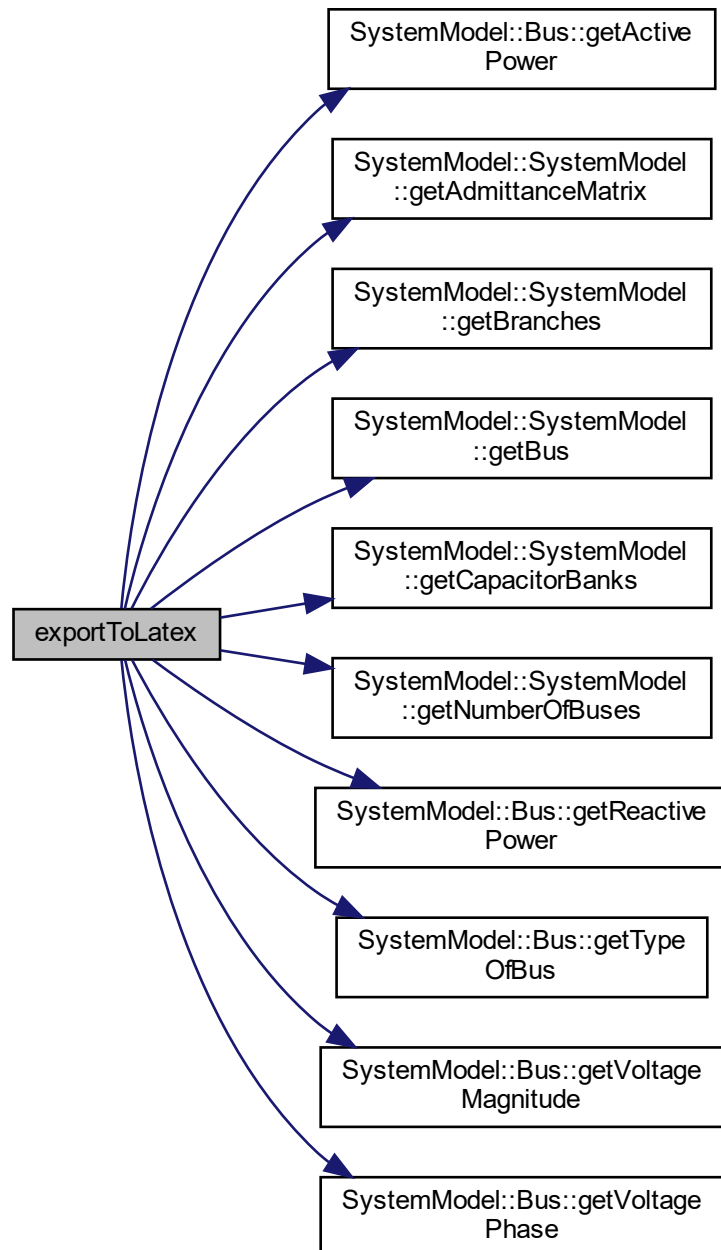
Exports `SystemModel` to the main.tex file

Parameters

SystemModel::SystemModel	System model
--	--------------

Returns

Here is the call graph for this function:



6.1.1.3 exportToTxt()

```
void exportToTxt (
    const char * filename,
    SystemModel::SystemModel s )
```

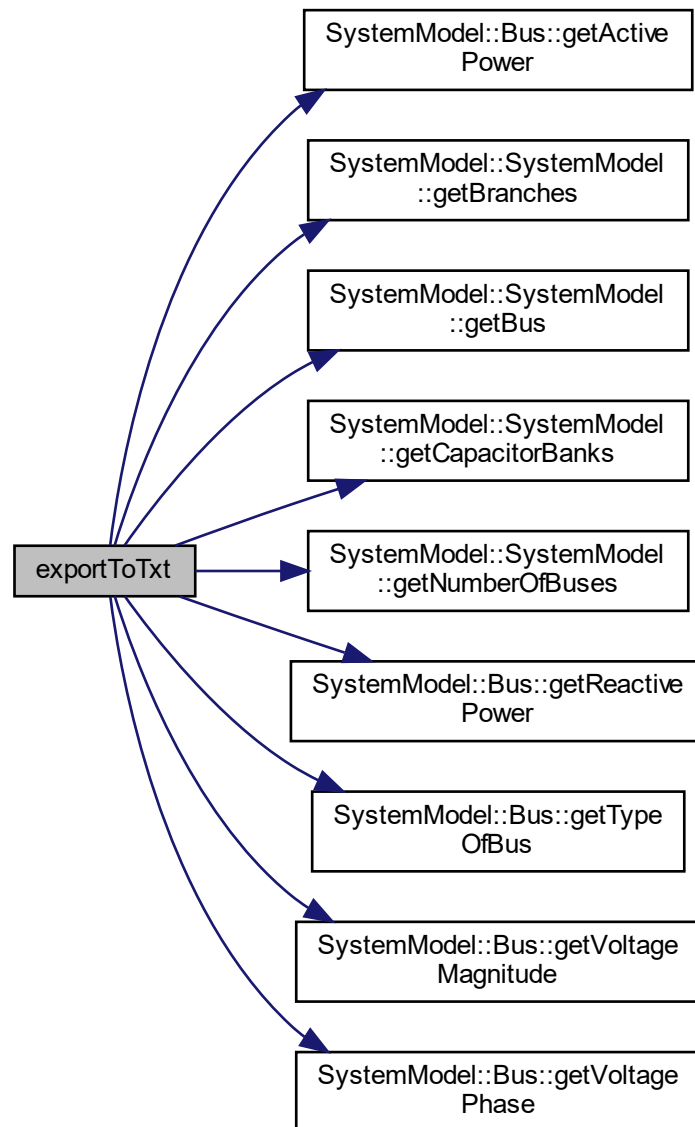
Exports [SystemModel](#) to the .tex file

Parameters

<i>const char*</i>	Name of the file
SystemModel::SystemModel	System model

Returns

Here is the call graph for this function:



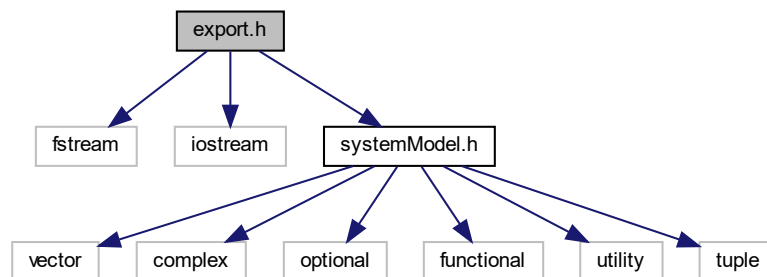
6.1.2 Variable Documentation

6.1.2.1 `eps`

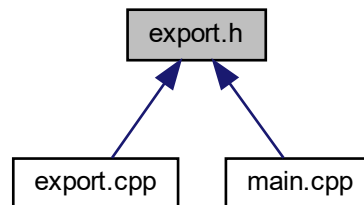
```
const double eps { 1e-10 }
```

6.2 export.h File Reference

```
#include <fstream>
#include <iostream>
#include "systemModel.h"
Include dependency graph for export.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void `exportToLatex` (`SystemModel::SystemModel` s)
Exports `SystemModel` to the `main.tex` file
- void `exportToHTML` (`SystemModel::SystemModel` s)
Exports `SystemModel` to the `main.html` file
- void `exportToTxt` (const char *filename, `SystemModel::SystemModel` s)
Exports `SystemModel` to the `.tex` file

6.2.1 Function Documentation

6.2.1.1 exportToHTML()

```
void exportToHTML (
    SystemModel::SystemModel s )
```

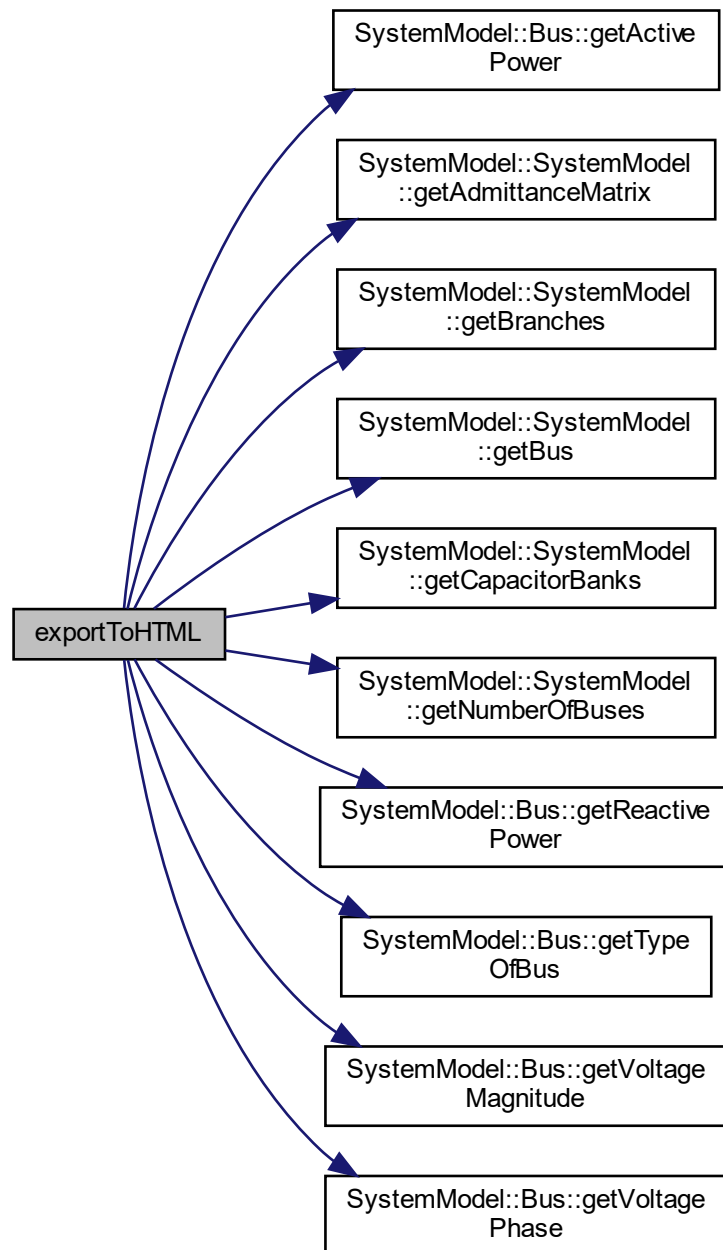
Exports [SystemModel](#) to the main.html file

Parameters

SystemModel::SystemModel	System model
--	--------------

Returns

Here is the call graph for this function:



6.2.1.2 exportToLatex()

```
void exportToLatex (  
    SystemModel::SystemModel s )
```

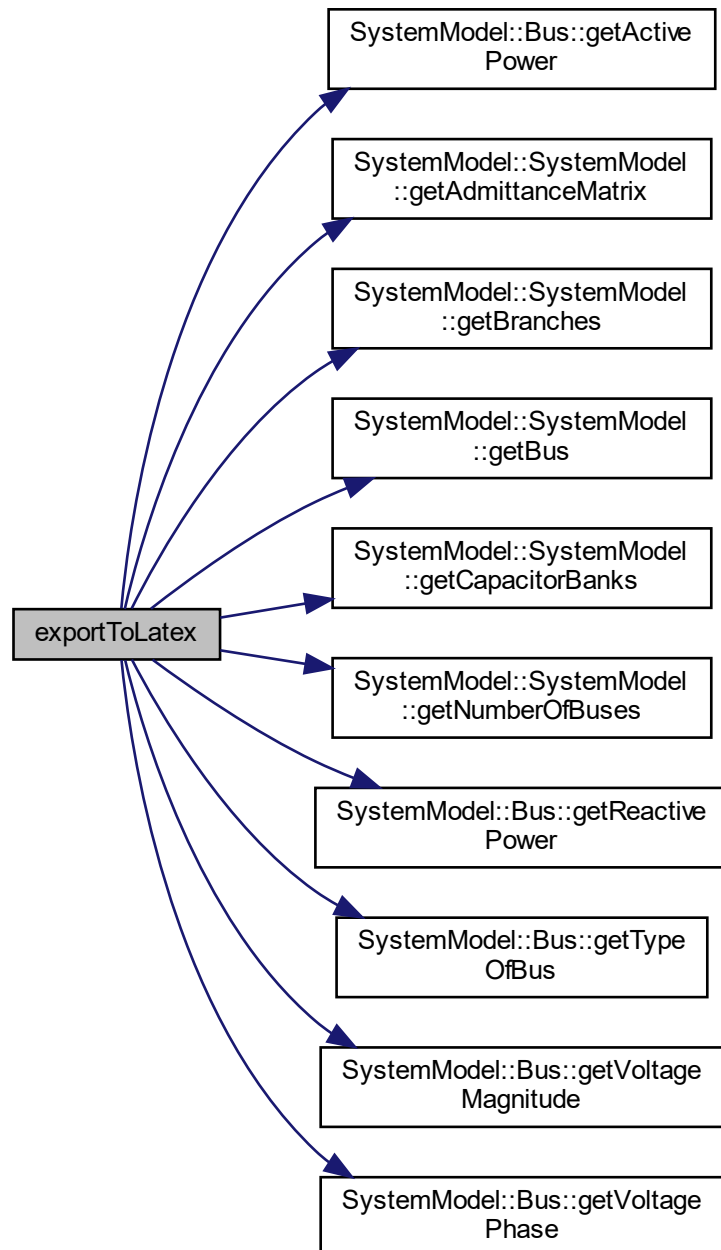
Exports `SystemModel` to the main.tex file

Parameters

SystemModel::SystemModel	System model
--	--------------

Returns

Here is the call graph for this function:



6.2.1.3 exportToTxt()

```
void exportToTxt (
    const char * filename,
    SystemModel::SystemModel s )
```

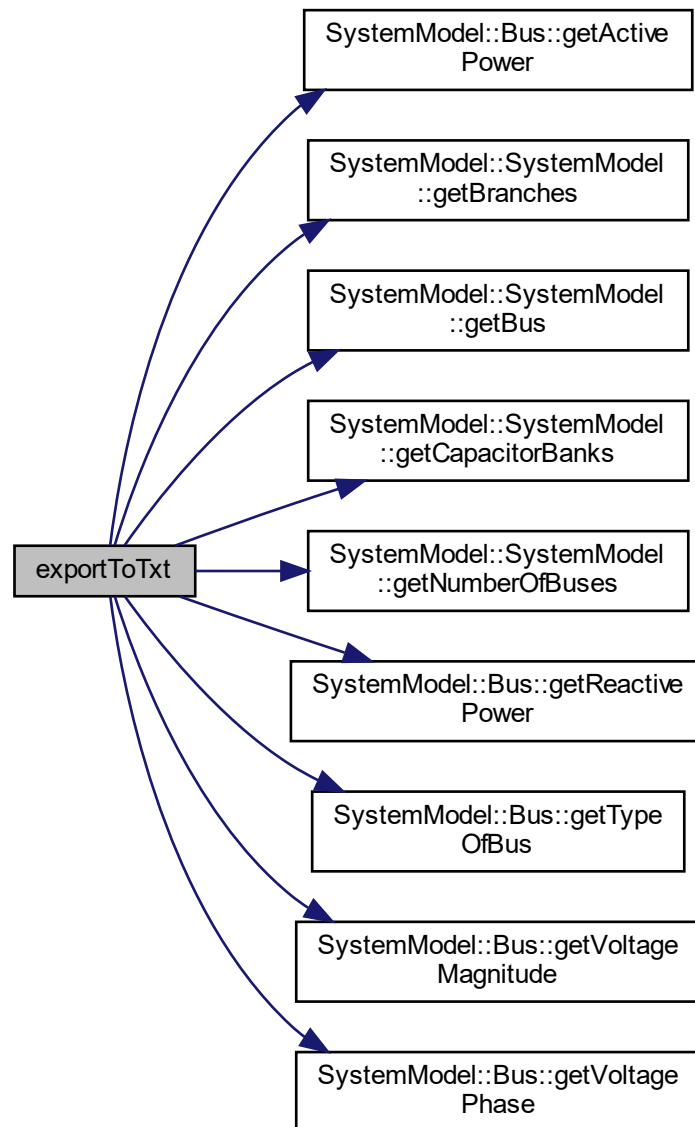
Exports [SystemModel](#) to the .tex file

Parameters

<i>const char*</i>	Name of the file
SystemModel::SystemModel	System model

Returns

Here is the call graph for this function:



6.3 export.h

[Go to the documentation of this file.](#)

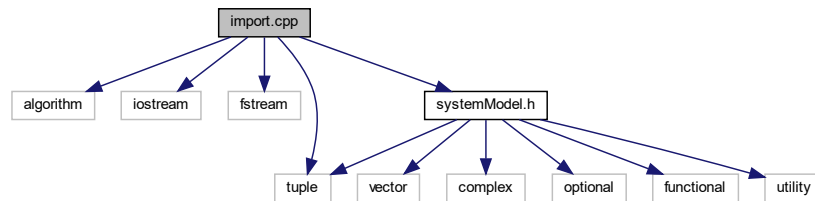
```

1 #pragma once
2 #include <fstream>
3 #include <iostream>
4 #include "systemModel.h"
5
6 void exportToLatex(SystemModel::SystemModel s);
7
8 void exportToHTML(SystemModel::SystemModel s);
9
10 void exportToTxt(const char* filename, SystemModel::SystemModel s);
  
```


6.4 import.cpp File Reference

```
#include <algorithm>
#include <iostream>
#include <fstream>
#include <tuple>
#include "systemModel.h"
```

Include dependency graph for import.cpp:



Functions

- void [importFromTxt](#) (const char *filename, [SystemModel::SystemModel](#) &systemModel)
Imports [SystemModel](#) from the .txt file

6.4.1 Function Documentation

6.4.1.1 importFromTxt()

```
void importFromTxt (
    const char * filename,
    SystemModel::SystemModel & systemModel )
```

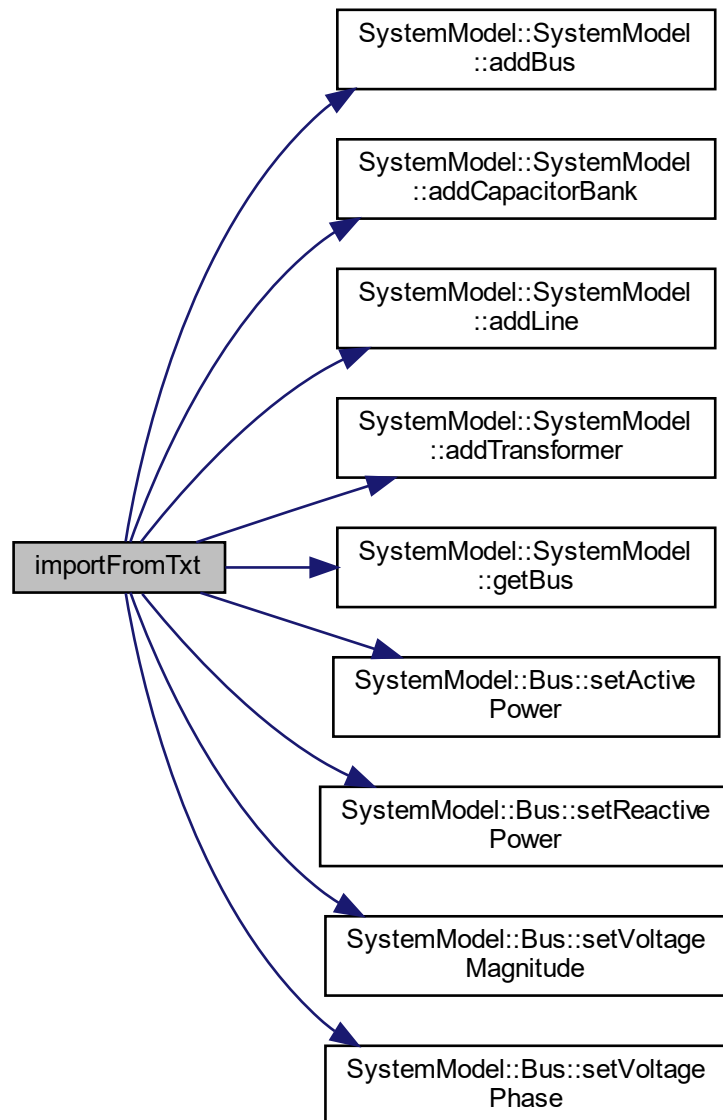
Imports [SystemModel](#) from the .txt file

Parameters

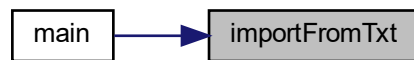
<i>const char*</i>	Name of the file
SystemModel::SystemModel	System model

Returns

Here is the call graph for this function:



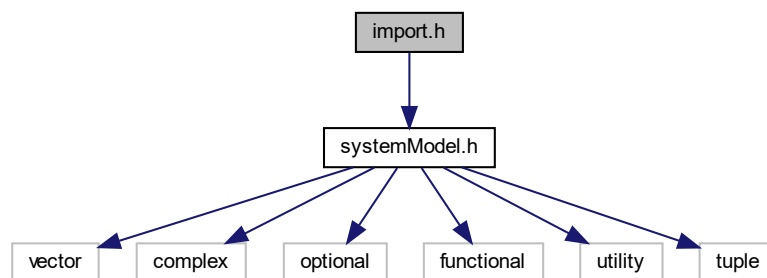
Here is the caller graph for this function:



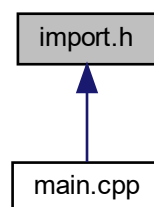
6.5 import.h File Reference

```
#include "systemModel.h"
```

Include dependency graph for import.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `importFromTxt` (const char *filename, [SystemModel::SystemModel](#) &s)
Imports [SystemModel](#) from the .txt file

6.5.1 Function Documentation

6.5.1.1 importFromTxt()

```
void importFromTxt (
    const char * filename,
    SystemModel::SystemModel & systemModel )
```

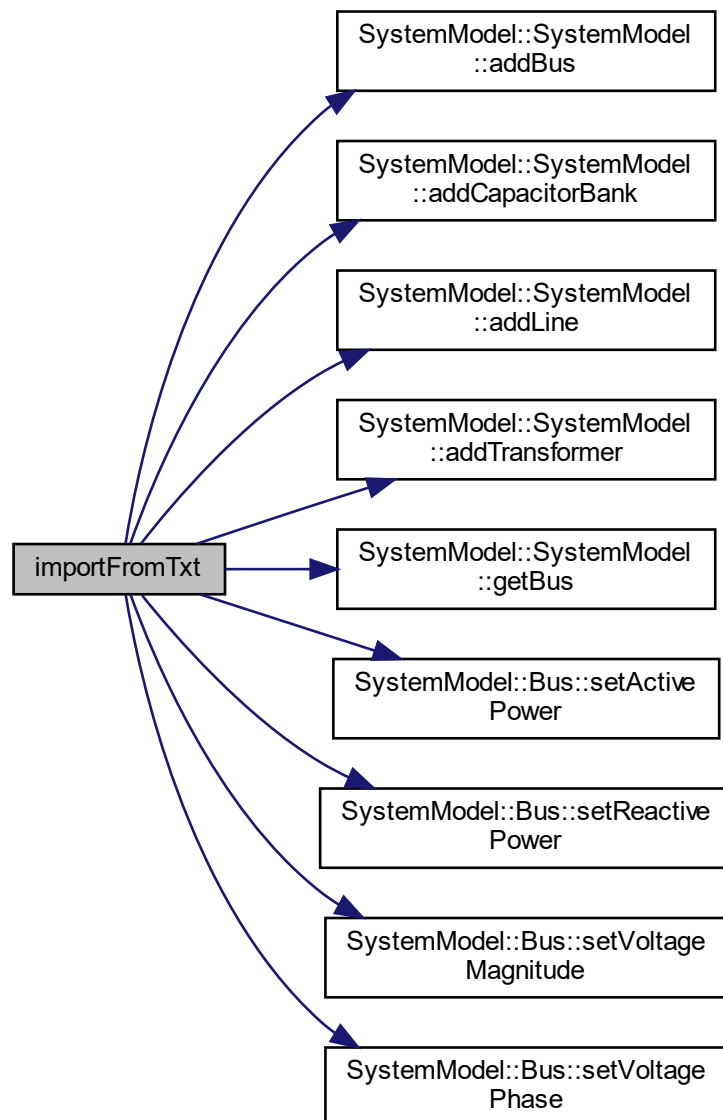
Imports [SystemModel](#) from the .txt file

Parameters

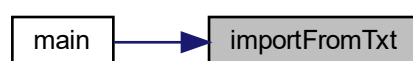
<i>const char*</i>	Name of the file
SystemModel::SystemModel	System model

Returns

Here is the call graph for this function:



Here is the caller graph for this function:



6.6 import.h

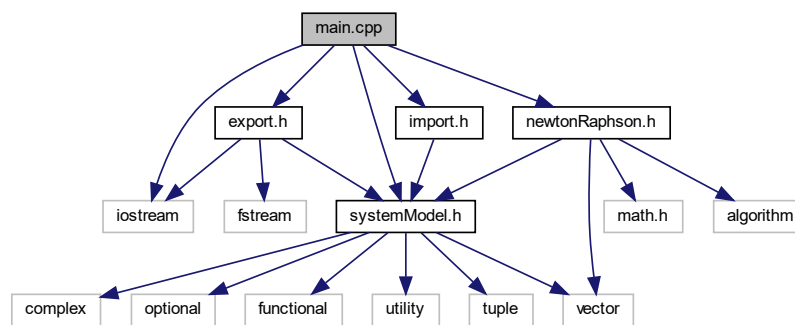
[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #include "systemModel.h"
4
5
6 void importFromTxt(const char* filename, SystemModel::SystemModel& s);
```

6.7 main.cpp File Reference

```
#include <iostream>
#include "systemModel.h"
#include "newtonRaphson.h"
#include "export.h"
#include "import.h"
```

Include dependency graph for main.cpp:



Functions

- int [main](#) ()

Variables

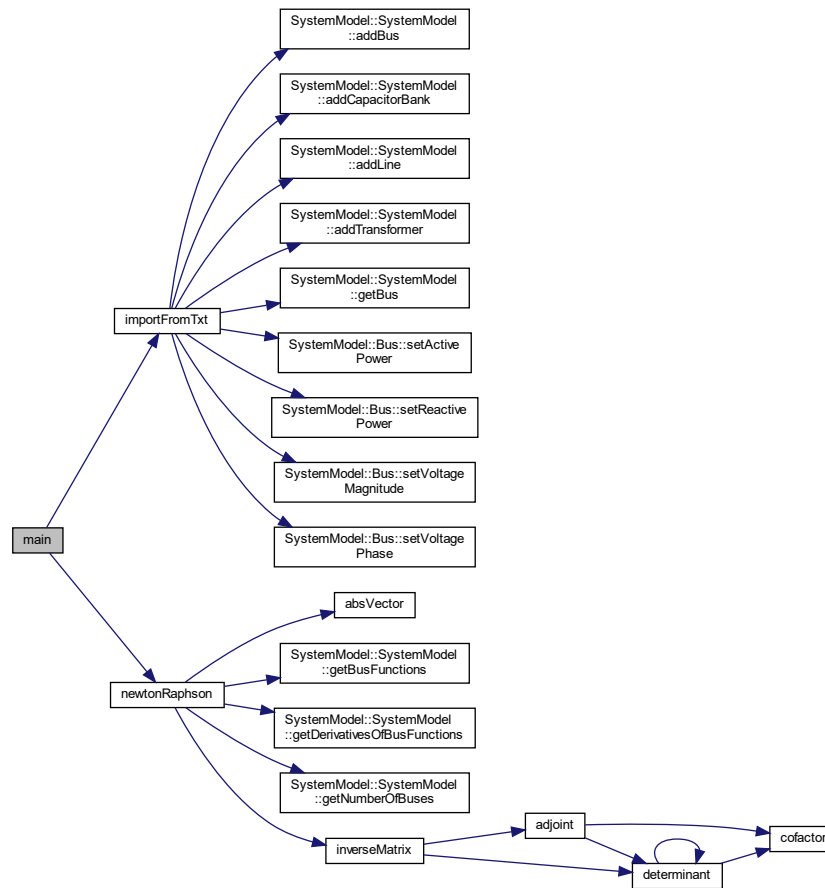
- const double [eps](#) { 1e-10 }

6.7.1 Function Documentation

6.7.1.1 main()

```
int main ( )
```

Here is the call graph for this function:



6.7.2 Variable Documentation

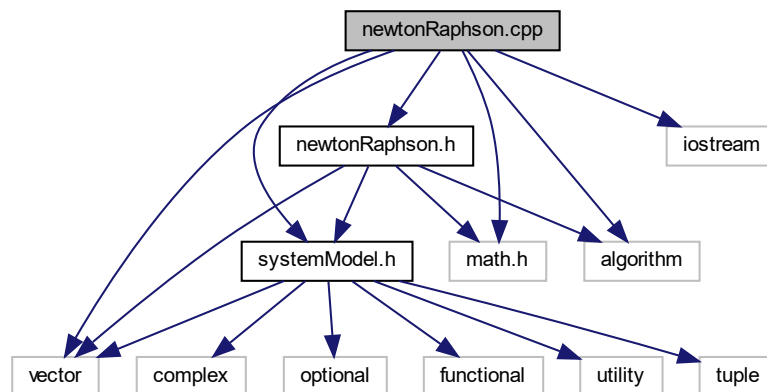
6.7.2.1 eps

```
const double eps { 1e-10 }
```

6.8 newtonRaphson.cpp File Reference

```
#include "systemModel.h"
#include "newtonRaphson.h"
#include <math.h>
#include <algorithm>
#include <iostream>
#include <vector>
```

Include dependency graph for newtonRaphson.cpp:



Functions

- `template<typename T >`
`std::vector< T > operator* (const std::vector< std::vector< T > > &matrix, const std::vector< T > &vector)`
Matrix and vector product operator overload
- `template<typename T >`
`std::vector< std::vector< T > > operator- (const std::vector< std::vector< T > > &matrix)`
Unary minus sign matrix operator overload
- `std::vector< double > absVector (const std::vector< double > &vec)`
Absolute value of vector
- `void cofactor (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &t, int p, int q, int n)`
Cofactor of the matrix
- `double determinant (std::vector< std::vector< double > > matrix, int n)`
Determinant of matrix
- `void adjoint (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &adj)`
Adjoint matrix
- `std::vector< std::vector< double > > inverseMatrix (const std::vector< std::vector< double > > &matrix, double eps=1e-10)`
Inverse matrix
- `int newtonRaphson (SystemModel::SystemModel sm, int maxNumberOfIter, double eps, std::vector< double > x0, std::vector< double > &x, double &err, int &iter)`
Newton Raphson method

6.8.1 Function Documentation

6.8.1.1 absVector()

```
std::vector< double > absVector (
    const std::vector< double > & vec )
```

Absolute value of vector

Parameters

<code>std::vector<double></code>	Vector of double elements
--	---------------------------

Returns

Absolute value of elements in the argument vector

Here is the caller graph for this function:



6.8.1.2 adjoint()

```
void adjoint (
    const std::vector< std::vector< double > > & matrix,
    std::vector< std::vector< double > > & adj )
```

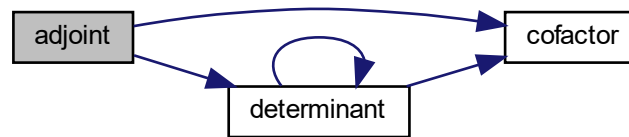
Adjoint matrix

Parameters

<code>std::vector<std::vector<double>></code>	Matrix to get adjoint from
<code>std::vector<std::vector<double>></code>	Referece to adjoint matrix

Returns

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.3 cofactor()

```

void cofactor (
    const std::vector< std::vector< double > > & matrix,
    std::vector< std::vector< double > > & t,
    int p,
    int q,
    int n )
  
```

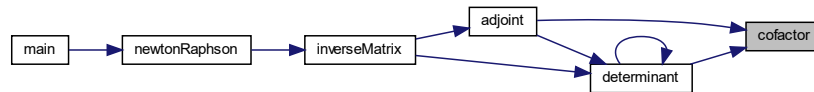
Cofactor of the matrix

Parameters

<i>std::vector<std::vector<double>></i>	Matrix to get cofactor from
<i>std::vector<std::vector<double>></i>	Cofactor matrix
<i>int</i>	Row of the cofactor that needs to be found
<i>int</i>	Column of the cofactor that needs to be found
<i>int</i>	Size of square matrix

Returns

Here is the caller graph for this function:



6.8.1.4 determinant()

```
double determinant (
    std::vector< std::vector< double > > matrix,
    int n )
```

Determinant of matrix

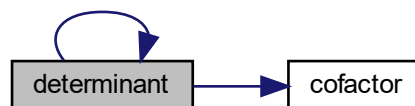
Parameters

<code>std::vector<std::vector<double>></code>	Matrix to get determinant from
<code>int</code>	Size of square matrix

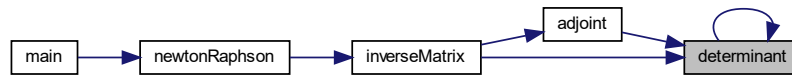
Returns

Double value of determinant

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.5 inverseMatrix()

```

std::vector< std::vector< double > > inverseMatrix (
    const std::vector< std::vector< double > > & matrix,
    double eps = 1e-10 )

```

Inverse matrix

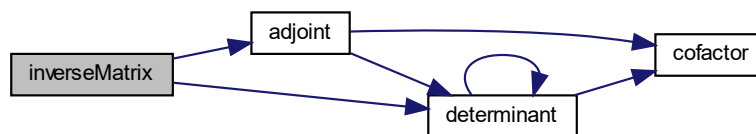
Parameters

<code>std::vector<std::vector<double>></code>	Matrix to get inverse from
<code>double</code>	Sinuglarity check

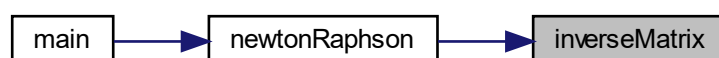
Returns

Matrix that is inverse from the first argument

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.6 newtonRaphson()

```
int newtonRaphson (
    SystemModel::SystemModel sm,
    int maxNumberOfIter,
    double eps,
    std::vector< double > x0,
    std::vector< double > & x,
    double & err,
    int & iter )
```

Newton Raphson method

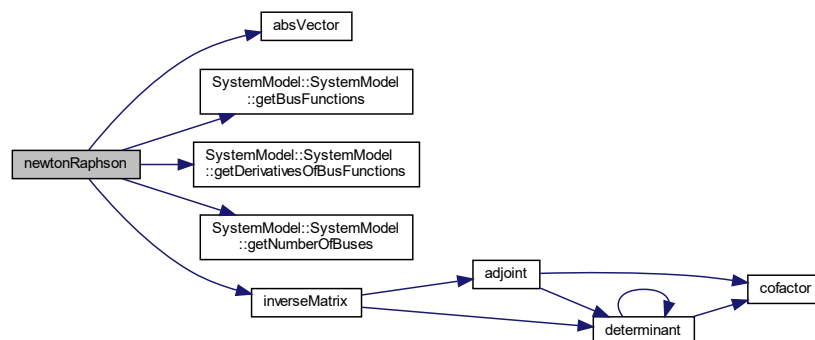
Parameters

SystemModel::SystemModel	System model
<i>int</i>	Maximum number of iterations
<i>double</i>	Maximum tolerance
<i>std::vector<double></i>	Starting solution vector
<i>std::vector<double></i>	Reference to solution vector
<i>double</i>	Reference to tolerance achieved
<i>int</i>	Reference to number of iterations preformed

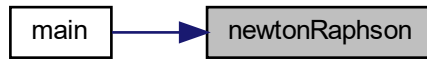
Returns

Int value that shows if the system converges or not. Returns 1 if converges, returns 0 if it does not

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.7 operator*()

```

template<typename T >
std::vector< T > operator* (
    const std::vector< std::vector< T > > & matrix,
    const std::vector< T > & vector )
  
```

Matrix and vector product operator overload

Parameters

<i>std::vector<std::vector<T>></i>	Vector of vector type
<i>std::vector<T></i>	Vector type

Returns

Vector that is the result of matrix and vector product

6.8.1.8 operator-()

```

template<typename T >
std::vector< std::vector< T > > operator- (
    const std::vector< std::vector< T > > & matrix )
  
```

Unary minus sign matrix operator overload

Parameters

<i>std::vector<std::vector<T>></i>	Matrix of type
--	----------------

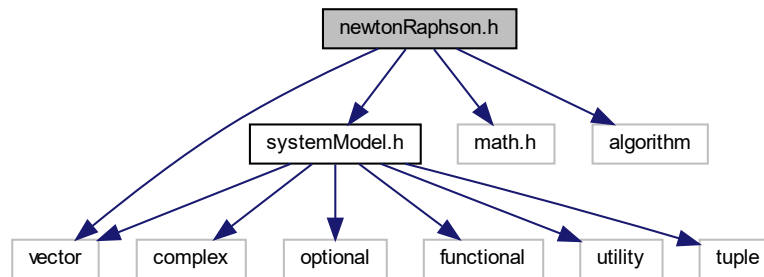
Returns

Reverse sign elements of matrix

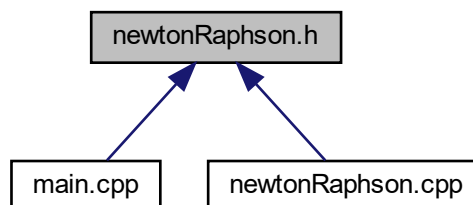
6.9 newtonRaphson.h File Reference

```
#include "systemModel.h"
#include <vector>
#include <math.h>
#include <algorithm>
```

Include dependency graph for newtonRaphson.h:



This graph shows which files directly or indirectly include this file:



Functions

- `template<typename T>`
`std::vector< T > operator* (const std::vector< std::vector< T > > &matrix, const std::vector< T > &vector)`
Matrix and vector product operator overload
- `template<typename T>`
`std::vector< std::vector< T > > operator- (const std::vector< std::vector< T > > &matrix)`
Unary minus sign matrix operator overload
- `std::vector< double > absVector (const std::vector< double > &vec)`
Absolute value of vector
- `void cofactor (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &t,`
`int p, int q, int n)`
Cofactor of the matrix

- double [determinant](#) (std::vector< std::vector< double > > matrix, int n)
Determinant of matrix
- void [adjoint](#) (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &adj)
Adjoint matrix
- std::vector< std::vector< double > > [inverseMatrix](#) (const std::vector< std::vector< double > > &matrix, double [eps](#))
Inverse matrix
- int [newtonRaphson](#) (SystemModel::SystemModel sm, int maxNumberOfIter, double [eps](#), std::vector< double > > x0, std::vector< double > &x, double &err, int &iter)
Newton Raphson method

6.9.1 Function Documentation

6.9.1.1 absVector()

```
std::vector< double > absVector (
    const std::vector< double > & vec )
```

Absolute value of vector

Parameters

<code>std::vector<double></code>	Vector of double elements
--	---------------------------

Returns

Absolute value of elements in the argument vector

Here is the caller graph for this function:



6.9.1.2 adjoint()

```
void adjoint (
    const std::vector< std::vector< double > > & matrix,
    std::vector< std::vector< double > > & adj )
```

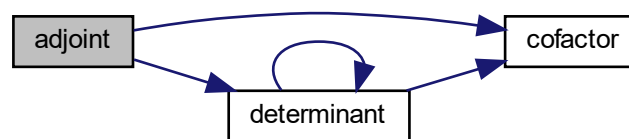
Adjoint matrix

Parameters

<code>std::vector<std::vector<double>></code>	Matrix to get adjoint from
<code>std::vector<std::vector<double>></code>	Referece to adjoint matrix

Returns

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.1.3 cofactor()

```

void cofactor (
    const std::vector< std::vector< double > > & matrix,
    std::vector< std::vector< double > > & t,
    int p,
    int q,
    int n )

```

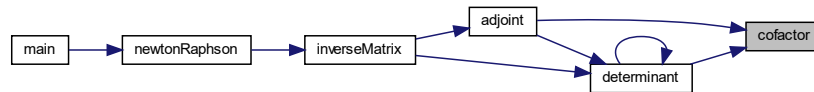
Cofactor of the matrix

Parameters

<code>std::vector<std::vector<double>></code>	Matrix to get cofactor from
<code>std::vector<std::vector<double>></code>	Cofactor matrix
<code>int</code>	Row of the cofactor that needs to be found
<code>int</code>	Column of the cofactor that needs to be found
<code>int</code>	Size of square matrix

Returns

Here is the caller graph for this function:



6.9.1.4 determinant()

```
double determinant (
    std::vector< std::vector< double > > matrix,
    int n )
```

Determinant of matrix

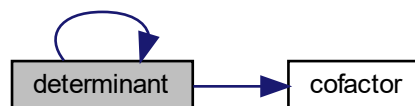
Parameters

<code>std::vector<std::vector<double>></code>	Matrix to get determinant from
<code>int</code>	Size of square matrix

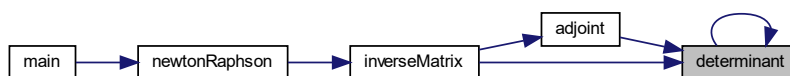
Returns

Double value of determinant

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.1.5 inverseMatrix()

```

std::vector< std::vector< double > > inverseMatrix (
    const std::vector< std::vector< double > > & matrix,
    double eps = 1e-10 )

```

Inverse matrix

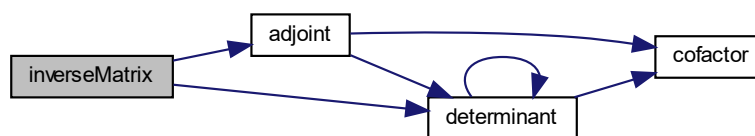
Parameters

<code>std::vector<std::vector<double>></code>	Matrix to get inverse from
<code>double</code>	Sinuglarity check

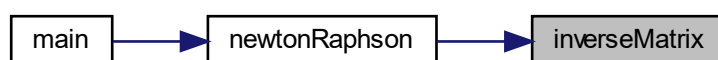
Returns

Matrix that is inverse from the first argument

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.1.6 newtonRaphson()

```
int newtonRaphson (
    SystemModel::SystemModel sm,
    int maxNumberOfIter,
    double eps,
    std::vector< double > x0,
    std::vector< double > & x,
    double & err,
    int & iter )
```

Newton Raphson method

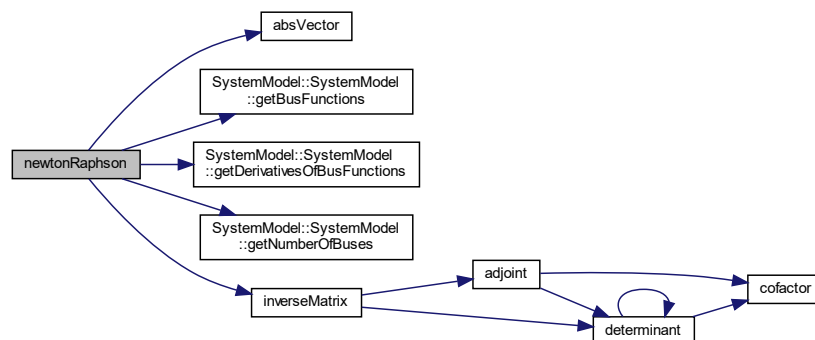
Parameters

SystemModel::SystemModel	System model
<i>int</i>	Maximum number of iterations
<i>double</i>	Maximum tolerance
<i>std::vector<double></i>	Starting solution vector
<i>std::vector<double></i>	Reference to solution vector
<i>double</i>	Reference to tolerance achieved
<i>int</i>	Reference to number of iterations preformed

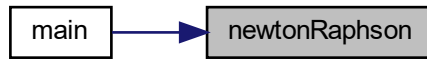
Returns

Int value that shows if the system converges or not. Returns 1 if converges, returns 0 if it does not

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.1.7 operator*()

```

template<typename T >
std::vector< T > operator* (
    const std::vector< std::vector< T > > & matrix,
    const std::vector< T > & vector )
  
```

Matrix and vector product operator overload

Parameters

<code>std::vector<std::vector<T>></code>	Vector of vector type
<code>std::vector<T></code>	Vector type

Returns

Vector that is the result of matrix and vector product

6.9.1.8 operator-()

```

template<typename T >
std::vector< std::vector< T > > operator- (
    const std::vector< std::vector< T > > & matrix )
  
```

Unary minus sign matrix operator overload

Parameters

<code>std::vector<std::vector<T>></code>	Matrix of type
--	----------------

Returns

Reverse sign elements of matrix

6.10 newtonRaphson.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include "systemModel.h"
3 #include <vector>
4 #include <math.h>
5 #include <algorithm>
6
7 template <typename T>
8 std::vector<T> operator *(const std::vector<std::vector<T>& matrix, const std::vector<T>& vector);
9
10 template <typename T>
11 std::vector<std::vector<T> operator -(const std::vector<std::vector<T>& matrix);
12
13 std::vector<double> absVector(const std::vector<double>& vec);
14
15 void cofactor(const std::vector<std::vector<double>& matrix, std::vector<std::vector<double>& t, int p,
16               int q, int n);
17
18 double determinant(std::vector<std::vector<double>& matrix, int n);
19
20 void adjoint(const std::vector<std::vector<double>& matrix, std::vector<std::vector<double>& adj);
21
22 std::vector<std::vector<double> inverseMatrix(const std::vector<std::vector<double>& matrix, double eps);
23
24 int newtonRaphson(SystemModel::SystemModel sm, int maxNumberOfIter, double eps, std::vector<double> x0,
25                  std::vector<double> x, double& err, int& iter);

```

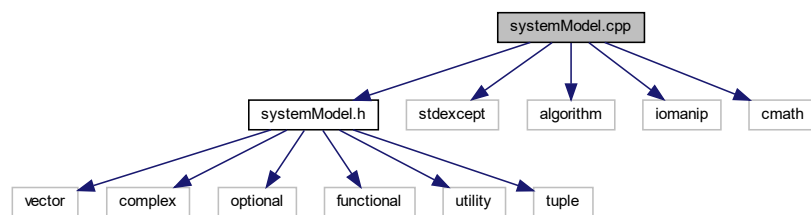
6.11 systemModel.cpp File Reference

```

#include "systemModel.h"
#include <stdexcept>
#include <algorithm>
#include <iomanip>
#include <cmath>

```

Include dependency graph for systemModel.cpp:



Macros

- #define PI 4 * std::atan(1.0)

6.11.1 Macro Definition Documentation

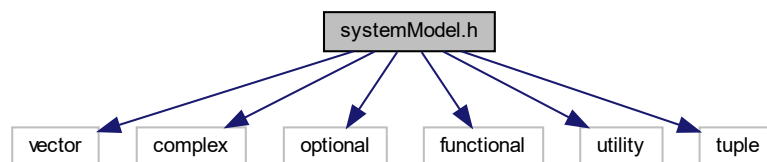
6.11.1.1 PI

```
#define PI 4 * std::atan(1.0)
```

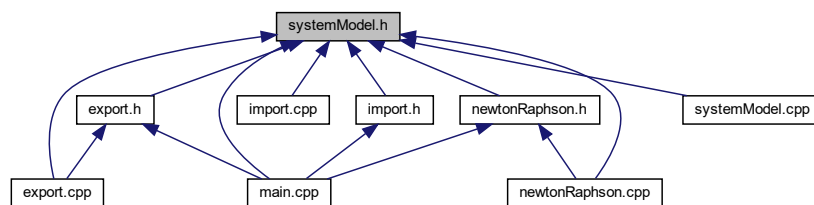
6.12 systemModel.h File Reference

```
#include <vector>
#include <complex>
#include <optional>
#include <functional>
#include <utility>
#include <tuple>
```

Include dependency graph for systemModel.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SystemModel::Bus](#)
- class [SystemModel::SystemModel](#)

Namespaces

- namespace [SystemModel](#)

Typedefs

- using `SystemModel::fi` = `std::pair< std::function< double(std::vector< double >>)>, std::function< double(std::vector< double >>)>>`
- using `SystemModel::dfidx` = `std::pair< std::vector< std::function< double(std::vector< double >>)>>, std::vector< std::function< double(std::vector< double >>)>>>`
- using `SystemModel::AdmittanceMatrix` = `std::vector< std::tuple< uint8_t, uint8_t, std::complex< double >>>`
- using `SystemModel::Branch` = `std::tuple< TypeOfBranch, uint8_t, uint8_t, double, double, double, double >`

Enumerations

- enum class `SystemModel::TypeOfBus` { `SystemModel::Slack`, `SystemModel::PV`, `SystemModel::PQ` }
- enum class `SystemModel::ThreePhaseLoadConfigurationsType` { `SystemModel::Star`, `SystemModel::GroundedStar`, `SystemModel::Delta` }
- enum class `SystemModel::TypeOfBranch` { `SystemModel::Line`, `SystemModel::Transformer` }

Functions

- `std::ostream & SystemModel::operator<<` (`std::ostream &stream`, `const SystemModel &systemModel`)
Output stream operator overload

6.13 systemModel.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include <vector>
3 #include <complex>
4 #include <optional>
5 #include <functional>
6 #include <utility>
7 #include <tuple>
8
9
10
11 namespace SystemModel {
12     enum class TypeOfBus { Slack, PV, PQ };
13
14
15
16     enum class ThreePhaseLoadConfigurationsType { Star, GroundedStar, Delta };
17
18
19
20     enum class TypeOfBranch { Line, Transformer };
21
22
23
24     using fi = std::pair<std::function<double(std::vector<double>>)>,
25         std::function<double(std::vector<double>>)>>;
26
27
28     using dfidx = std::pair<std::vector<std::function<double(std::vector<double>>)>>,
29         std::vector<std::function<double(std::vector<double>>)>>>;
30
31
32     using AdmittanceMatrix = std::vector<std::tuple<uint8_t, uint8_t, std::complex<double>>>;
33
34
35
36     using Branch = std::tuple<TypeOfBranch, uint8_t, uint8_t, double, double, double, double>;
37
38
39

```



```

40     class Bus {
41         TypeOfBus typeOfBus;
42         std::optional<double> voltageMagnitude;
43         std::optional<double> voltagePhase;
44         std::optional<double> activePower;
45         std::optional<double> reactivePower;
46     public:
47         Bus(TypeOfBus typeOfBus) : typeOfBus{ typeOfBus } {}
48
49         TypeOfBus getTypeOfBus() const {
50             return typeOfBus;
51         }
52
53         void setVoltageMagnitude(double voltageMagnitude);
54
55         void setVoltagePhase(double voltagePhase);
56
57         void setActivePower(double activePower);
58
59         void setReactivePower(double reactivePower);
60
61         std::optional<double> getVoltageMagnitude() const;
62
63         std::optional<double> getVoltagePhase() const;
64
65         std::optional<double> getActivePower() const;
66
67         std::optional<double> getReactivePower() const;
68     };
69
70
71     class SystemModel {
72         AdmittanceMatrix admittanceMatrix;
73         uint8_t numberOfBuses{};
74         std::vector<Bus> buses;
75         const uint8_t maxNumberOfBuses;
76         bool checkForConnectionBetweenToBuses(uint8_t busNumber1, uint8_t busNumber2) const;
77         std::vector<Branch> branches;
78         void addBranchToAdmittanceMatrix(uint8_t busNumber1, uint8_t busNumber2, double r, double x,
79         double g, double b);
80         std::vector<std::tuple<uint8_t, double, ThreePhaseLoadConfigurationsType>> capacitorBanks;
81         void addCapacitorBankToAdmittanceMatrix(uint8_t busNumber, double b,
82         ThreePhaseLoadConfigurationsType configurationType);
83         void recalculateAdmittanceMatrix();
84     public:
85         SystemModel(uint8_t maxNumberOfBuses) : maxNumberOfBuses{ maxNumberOfBuses } {}
86
87         AdmittanceMatrix getAdmittanceMatrix() const {
88             return admittanceMatrix;
89         }
90
91         uint8_t getNumberOfBuses() const {
92             return numberOfBuses;
93         }
94
95         Bus& getBus(uint8_t busNumber);
96
97         void addBus(TypeOfBus typeOfBus);
98
99         void addLoad(uint8_t busNumber, double activePower, double reactivePower);
100
101         void addLine(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b);
102
103         friend std::ostream& operator <<(std::ostream& stream, const SystemModel& systemModel);
104
105         void addGenerator(uint8_t busNumber, double voltageMagnitude, double activePower);
106
107         void addSlackGenerator(uint8_t busNumber, double voltageMagnitude, double voltagePhase);
108
109         bool hasSlackBeenAssigned() const;
110
111         void addTransformer(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g, double
112         b);
113
114         void addCapacitorBank(uint8_t busNumber, double b, ThreePhaseLoadConfigurationsType
115         configurationType);
116
117         fi getBusFunctions(uint8_t busNumber) const;
118
119         dfidx getDerivativesOfBusFunctions(uint8_t busNumber) const;
120
121         void removeBranch(uint8_t busNumber1, uint8_t busNumber2);
122
123         void changeLine(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b);
124
125         void changeTransformer(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g,

```

```
double b);
123
124     std::vector<Branch> getBranches() const {
125         return branches;
126     }
127
128     void removeBus(uint8_t busNumber);
129
130     void removeCapacitorBank(uint8_t busNumber);
131
132     void changeCapacitorBank(uint8_t busNumber, double b, ThreePhaseLoadConfigurationsType
configurationType);
133
134     std::vector<std::tuple<uint8_t, double, ThreePhaseLoadConfigurationsType> getCapacitorBanks()
const {
135         return capacitorBanks;
136     }
137 };
138
139
140
141     std::ostream& operator <<(std::ostream& stream, const SystemModel& systemModel);
142 }
```

Index

- absVector
 - newtonRaphson.cpp, [51](#)
 - newtonRaphson.h, [58](#)
- addBus
 - SystemModel::SystemModel, [18](#)
- addCapacitorBank
 - SystemModel::SystemModel, [19](#)
- addGenerator
 - SystemModel::SystemModel, [19](#)
- addLine
 - SystemModel::SystemModel, [20](#)
- addLoad
 - SystemModel::SystemModel, [20](#)
- addSlackGenerator
 - SystemModel::SystemModel, [21](#)
- addTransformer
 - SystemModel::SystemModel, [21](#)
- adjoint
 - newtonRaphson.cpp, [51](#)
 - newtonRaphson.h, [58](#)
- AdmittanceMatrix
 - SystemModel, [7](#)
- Branch
 - SystemModel, [8](#)
- Bus
 - SystemModel::Bus, [12](#)
- changeCapacitorBank
 - SystemModel::SystemModel, [22](#)
- changeLine
 - SystemModel::SystemModel, [22](#)
- changeTransformer
 - SystemModel::SystemModel, [23](#)
- cofactor
 - newtonRaphson.cpp, [52](#)
 - newtonRaphson.h, [59](#)
- Delta
 - SystemModel, [8](#)
- determinant
 - newtonRaphson.cpp, [53](#)
 - newtonRaphson.h, [60](#)
- dfidx
 - SystemModel, [8](#)
- eps
 - export.cpp, [36](#)
 - main.cpp, [49](#)
- export.cpp, [31](#)
- eps, [36](#)
- exportToHTML, [32](#)
- exportToLatex, [33](#)
- exportToTxt, [34](#)
- export.h, [37](#)
- exportToHTML, [37](#)
- exportToLatex, [39](#)
- exportToTxt, [40](#)
- exportToHTML
 - export.cpp, [32](#)
 - export.h, [37](#)
- exportToLatex
 - export.cpp, [33](#)
 - export.h, [39](#)
- exportToTxt
 - export.cpp, [34](#)
 - export.h, [40](#)
- fi
 - SystemModel, [8](#)
- getActivePower
 - SystemModel::Bus, [12](#)
- getAdmittanceMatrix
 - SystemModel::SystemModel, [23](#)
- getBranches
 - SystemModel::SystemModel, [23](#)
- getBus
 - SystemModel::SystemModel, [24](#)
- getBusFunctions
 - SystemModel::SystemModel, [25](#)
- getCapacitorBanks
 - SystemModel::SystemModel, [25](#)
- getDerivativesOfBusFunctions
 - SystemModel::SystemModel, [26](#)
- getNumberOfBuses
 - SystemModel::SystemModel, [27](#)
- getReactivePower
 - SystemModel::Bus, [12](#)
- getTypeOfBus
 - SystemModel::Bus, [13](#)
- getVoltageMagnitude
 - SystemModel::Bus, [13](#)
- getVoltagePhase
 - SystemModel::Bus, [14](#)
- GroundedStar
 - SystemModel, [8](#)
- hasSlackBeenAssigned
 - SystemModel::SystemModel, [27](#)

- import.cpp, 43
 - importFromTxt, 43
- import.h, 45
 - importFromTxt, 46
- importFromTxt
 - import.cpp, 43
 - import.h, 46
- inverseMatrix
 - newtonRaphson.cpp, 54
 - newtonRaphson.h, 61
- Line
 - SystemModel, 9
- main
 - main.cpp, 48
- main.cpp, 48
 - eps, 49
 - main, 48
- newtonRaphson
 - newtonRaphson.cpp, 55
 - newtonRaphson.h, 62
- newtonRaphson.cpp, 50
 - absVector, 51
 - adjoint, 51
 - cofactor, 52
 - determinant, 53
 - inverseMatrix, 54
 - newtonRaphson, 55
 - operator*, 56
 - operator-, 56
- newtonRaphson.h, 57
 - absVector, 58
 - adjoint, 58
 - cofactor, 59
 - determinant, 60
 - inverseMatrix, 61
 - newtonRaphson, 62
 - operator*, 63
 - operator-, 63
- operator<<
 - SystemModel, 9
 - SystemModel::SystemModel, 28
- operator*
 - newtonRaphson.cpp, 56
 - newtonRaphson.h, 63
- operator-
 - newtonRaphson.cpp, 56
 - newtonRaphson.h, 63
- PI
 - systemModel.cpp, 64
- PQ
 - SystemModel, 9
- PV
 - SystemModel, 9
- removeBranch
 - SystemModel::SystemModel, 27
- removeBus
 - SystemModel::SystemModel, 28
- removeCapacitorBank
 - SystemModel::SystemModel, 28
- setActivePower
 - SystemModel::Bus, 14
- setReactivePower
 - SystemModel::Bus, 15
- setVoltageMagnitude
 - SystemModel::Bus, 15
- setVoltagePhase
 - SystemModel::Bus, 16
- Slack
 - SystemModel, 9
- Star
 - SystemModel, 8
- SystemModel, 7
 - AdmittanceMatrix, 7
 - Branch, 8
 - Delta, 8
 - dfidx, 8
 - fi, 8
 - GroundedStar, 8
 - Line, 9
 - operator<<, 9
 - PQ, 9
 - PV, 9
 - Slack, 9
 - Star, 8
 - SystemModel::SystemModel, 18
 - ThreePhaseLoadConfigurationsType, 8
 - Transformer, 9
 - TypeOfBranch, 8
 - TypeOfBus, 9
- systemModel.cpp, 64
 - PI, 64
- systemModel.h, 65
 - SystemModel::Bus, 11
 - Bus, 12
 - getActivePower, 12
 - getReactivePower, 12
 - getTypeOfBus, 13
 - getVoltageMagnitude, 13
 - getVoltagePhase, 14
 - setActivePower, 14
 - setReactivePower, 15
 - setVoltageMagnitude, 15
 - setVoltagePhase, 16
- SystemModel::SystemModel, 17
 - addBus, 18
 - addCapacitorBank, 19
 - addGenerator, 19
 - addLine, 20
 - addLoad, 20
 - addSlackGenerator, 21
 - addTransformer, 21
 - changeCapacitorBank, 22

- changeLine, [22](#)
- changeTransformer, [23](#)
- getAdmittanceMatrix, [23](#)
- getBranches, [23](#)
- getBus, [24](#)
- getBusFunctions, [25](#)
- getCapacitorBanks, [25](#)
- getDerivativesOfBusFunctions, [26](#)
- getNumberOfBuses, [27](#)
- hasSlackBeenAssigned, [27](#)
- operator<<, [28](#)
- removeBranch, [27](#)
- removeBus, [28](#)
- removeCapacitorBank, [28](#)
- SystemModel, [18](#)

ThreePhaseLoadConfigurationsType

- SystemModel, [8](#)

Transformer

- SystemModel, [9](#)

TypeOfBranch

- SystemModel, [8](#)

TypeOfBus

- SystemModel, [9](#)